OOD Assignment 4 – 90 Points – UML/Java OO System Design and Development

This assignment is intended to be done by you as a solo effort. The goal is again for you to extend your previous code from assignments 2 and 3 with design patterns, UML, and Java for additional functionality. This will be the last extension to this code base. You should use a Java 8 or later environment to develop this code (I recommend Java 17). You may use all or part of my example code from assignment 2 for this project, but please clearly indicate in comments if you do.

Overall Assignment – Zoo Simulation Extended Again

We will continue to simulate the zoo from assignment 2 and 3 in this assignment, all functionality previously provided should continue to work. This version of the simulation will introduce a new subclass – a new Staff subclass called Inspector – and a number of new behaviors and structural changes.

Implementing a Factory: You should implement a Factory pattern of code for either Animals or Staff. Each concrete subclass of Animal children (like Lion or Rhino, for instance) or of Staff children (Vendors, Handlers, etc.) should be created by a concrete factory creator object. Instead of your application making a call to new to create these new object instances, you should have a reference to factory objects that can provide the instances. This is not intended to be a simple factory OR abstract factory implementation. The implementation of your factory (and all patterns) should be clearly commented in your code for ease of grading.

Implementing a Decorator. Extend the functionality of your Toys shop by adding the following sales actions when the shop uses its sellitems method. When the Toys shop sells an item, it will also attempt to add the following items to the sale, using a concrete decorator method call for each additional item type:

- Sell Gift Wrapping (25% chance of adding this to a sale)
- Sell Gift Bag(s) (10% chance of adding one bag, 10% chance of adding two bags to a sale)
- Sell Gift Cards (20% chance of adding 1-3 (randomly determined) cards to a sale)

Be sure to announce any extra sales and include extra sales in the store's sales for the day.

Implementing Command: At the end of each day, the new Inspector will examine the Zoo. The inspector will randomly issue 1 to 4 commands each day. These commands should be created as command objects inheriting from an abstract command class or interface. When the command is executed, you can decide if other elements need to act as invokers or receivers, or if you want the command to directly request a specific action. The commands an Inspector may issue are as follows:

- Inspect enclosure: Inspector will want to know how many animals of a kind are in a random enclosure:
  - Typical message: The Inspector finds there are 2 animals in the Rhino enclosure.
- Inspect shop: Inspector will want to know sales information for a random shop.
  - Typical message: The Inspector finds the Drinks shop has sold $45.00 today.
- Inspect hospital: Inspector will want to know how many animals are currently in the hospital.
  - Typical message: The Inspector finds there are 5 animals in the hospital at present.
- Inspect animal: Inspector will look at a random animal to determine name, size, health, and location.
  - Typical message: The Inspector finds Leo the Lion is Medium sized, healthy, and in the Lion enclosure.

The Zoo simulation will run for 30 days as before. All other Zoo events from Staff will continue to occur and be printed to the console. Display the start and end of each numbered day in the console.

Assignment 3 Part 1 – UML designs for the Zoo Simulation

Using UML tools discussed in class, create the following UML diagrams.

1. **A UML Class Diagram** – This UML class diagram should include all classes, abstract classes, or interface definitions you feel you should use to design the Zoo simulation. The diagram should be complete, modeling all classes in use in the application. The diagram should clearly indicate any aggregation, association, or inheritance relationships, including any required multiplicity. In class elements of the diagram, identify key attributes and methods, including accessibility. (Usually this does not include constructors.) **You must highlight in the UML diagram where you are implementing Decorator, Factory, and Command patterns.**
2. **A Proposal for Project 5/6/7:**
   Proposal must include
   a. Project Title
   b. Project team member names (1, 2, or 3 students allowed)
   c. Project Description – must include use of a user interface, a data store, and some functionality being performed for the users
   d. Expected Pattern implementations – You must select at least four patterns you expect to apply to the project
   e. Language choice (including any known libraries or frameworks to be used)
   f. List of 2 to 4 functional elements per team member (ex. Login screen, Game piece graphics, etc.)

   It is understood that all these elements are subject to some changes when the actual project design and development begins, but this represents your initial intention for the work. The submitted proposal from a team of two or three can be the same information in each submission.

Please include your name on the submitted materials.

Submission Guidelines – Part 1

I would prefer that the diagrams are made with an electronic drawing tool for clarity. If you must hand draw the diagram (on a whiteboard or on paper) it must be complete and legible. The diagram should be turned in as a PDF. Most tools you use can print their output to a PDF, if necessary, you can break large diagrams into two or more pages. The important part is that what you turn in is readable and remember to put in your name!

Assignment 3 Part 2 – Java Code for the Zoo Simulation

Carefully consider the construction of your Java code based on your OO Design work. Implement the code required in Java to create, initialize, and run the Zoo simulation for 30 days. The code you write for the simulation should identify code related to the three added patterns: **Factory, Decorator, Command**. Clearly comment on at these implementations in your code so that they stand out when code is reviewed.

In addition, all code should be appropriately commented. Any .java files should include a *header block* with the author's name, the class, the program purpose, and the last revision date in the comments. Each *class/interface* and significant *method* in the code should be commented on as well. You do not have to use Javadoc formats for these comments, but you certainly can.

Capture ALL output from the simulation to the console (by cut/paste to a text file or optionally by directly writing to a text file). Also, update the UML class diagram from Part 1 to represent the final outcome of your Java code. Note any changes on the diagram since Part 1 as an annotation on the diagram itself.

If you use code in whole or in part from any external source, say Stack Overflow for example, you must cite your original source and include the URL in your comments. Never submit code copied from another source without providing appropriate citations.

Code in the Java main function should be limited to basic object instantiation and execution. No large blocks of procedural code or logic should be in Java main. All operations and information should be handled between appropriate objects of the solution classes.

Submission Guidelines – Part 2

Submit the following files to CougarVIEW for Part 2:

- any .java files required to run your program
- one text file with captured output from running the program for 30 simulated days
- your updated UML class diagram from Part 1 (as a PDF)

Grading Rubric:

Assignment 3 is worth 90 points total.

Part 1 (UML Diagram & Proposal) is 30 points and is due on Friday October 13 at 11:59 PM

- Each part is worth 15 points. A thorough design that meets all the requirements for class content and UML annotations will earn full points. Severe misses in UML syntax or incomplete design implementations will be penalized. The proposal must include all requested sections for full points.

Part 2 (Java Code & Results) is worth 60 points and is due on Friday October 20 at 11:59 PM

- (20 points) The code will be reviewed for good OO design – cohesive classes with clear logical connectivity and well named methods. Procedural elements or code duplication will be penalized. Code should be thoroughly commented as requested.
- (10 points) The implementing of the three added patterns **Command, Decorator, and Factory** should be clearly identified in comments.
- (20 points) The output from code execution should show all expected object actions and flow including new behaviors and objects. Missing elements will be penalized.
- (10 points) The updated UML Class Diagram should clearly show any changes since part 1, and an annotation on the drawing should describe the major changes. If no changes were made, this should be clearly annotated on the drawing as well.

Overall Project Guidelines

Assignments will be accepted late per the published policy from the syllabus:

- Up to two days late from posted submission date/time: -5% grade penalty
- Up to an additional three days late from posted submission date/time: -15% grade penalty
- Assignments will not be accepted after 5 days from posted submission date/time

Use office/student hours and e-mail to reach the instructor regarding homework/project questions, or if you have issues in completing the assignment for any reason.